一种基于 DQN 的无人驾驶任务卸载策略 *

王　锦，张新有†

(西南交通大学 计算机与人工智能学院，成都 611756)

摘　要：无人驾驶汽车由于其有限的电池寿命和计算能力，难以在保证续航的前提下满足一些时延敏感任务或密集任务的处理需求。为解决该问题，在移动边缘计算(mobile edge computing，MEC)的背景下，提出了一种基于深度 Q 网络(deep Q-network，DQN)的无人驾驶任务卸载策略。首先，定义了一个基于任务优先级的“车-边-云”协同任务卸载模型，其需要通过联合优化车辆计算能力与任务卸载策略以获取系统最小延迟和能耗。由于该问题是个混合整数非线性规划问题，因此分两步对其进行求解—通过数学推导得出了最优车辆计算能力的解析解，之后在其数值固定条件下，基于 DQN 算法获得了任务最佳卸载策略。最后，综合 SUMO、Pytorch 和 Python 等工具建立了仿真模型，比较了 DQN 算法和其他三种算法在任务负载、MEC 服务器计算能力以及能耗权重系数变化情况下的性能，实验结果验证了所提策略的可行性和优越性。

关键词：无人驾驶；移动边缘计算；任务卸载；深度 Q 网络；移动性

中图分类号：TP311　　doi: 10.19734/j.issn.1001-3695.2022.02.0043

DQN-based driverless task offloading policy

Wang Jin, Zhang Xinyou†

(*School of Computing & Artificial Intelligence*, *Southwest Jiaotong University*, *Chengdu* 611756, *China*)

Abstract: Due to its limited battery life and computing power, driverless cars are difficult to meet the processing needs of some delay-sensitive tasks or intensive tasks while ensuring battery life. To solve this problem, in the context of Mobile Edge Computing (MEC) , this paper proposed an driverless task offloading policy based on deep Q-network (DQN) . First, this paper defined a "vehicle-edge-cloud" cooperative task offloading model based on task priority, which needs to jointly optimize the computing power of the vehicle and the task offloading policy to obtain the minimum delay and energy consumption of the system. Since the problem is a mixed-integer nonlinear programming problem and is NP-hard, this paper solved it in two steps—the first step obtained the analytical solution for the optimal computation power of vehicle through mathematical derivation, and then, under the fixed numerical value condition, the DQN algorithm obtained the optimal offloading strategy of the task. Finally, this paper established a simulation model by integrating tools such as SUMO, Pytorch and Python. This paper compared the DQN algorithm and the other three algorithms under different task loads, MEC server computation powers and energy consumption weight co-efficients. The experimental results verify the feasibility and superiority of the proposed policy.

Key words: driverless cars; mobile edge computing; task offloading; deep Q-network; mobility

0　引言

近年来，无人驾驶汽车因其便利性和行驶效率受到了业界和学术界的广泛关注[1,2]。通过部署路边单元(RSU)等智能基础设施，车载用户可以获得路径规划、导航、车载娱乐等多种服务[3,4]。但是无人车本地搭载的设备计算能力较弱且电量有限，因此，目前大部分车辆将计算任务卸载到云端进行处理[5]。云服务器可以提供大量的计算资源，但较高的传输时延使时延敏感型应用的服务质量下降，甚至导致交通事故的发生。

MEC 的出现为无人驾驶技术的发展提供了切实有效的解决方案。通过在 RSU 旁部署 MEC 服务器，将计算资源、存储资源和通信资源分布到靠近用户的网络边缘侧[6]，从而降低了传输时延。但是如果将所有任务都卸载到 MEC 服务器上，容易使 MEC 服务器出现过载的状态。因此，无人车本地以及云服务器依然需要处理部分计算任务。

通过任务卸载，将计算密集型的任务迁移到 MEC 服务器上，一方面可以缓解无人车资源紧张的情况，减少无人车的能耗，增强其续航能力[7]，另一方面可以提高任务处理效率，减少任务完成时延。本文引入三层网络架构，车辆层、边缘层和云层，为资源受限的车辆提供了多元的卸载选择。无人车可以根据实际情况选择在本地处理自身的任务，也可以将任务卸载到 MEC 服务器或云服务器进行处理。

车-边-云架构下的任务卸载问题可以归结为资源受限条件下平衡车辆、MEC 服务器和云服务器的任务负载以获得最小时延和能耗的联合优化问题。出于对算法稳定性和可靠性等方面的考虑，本文采用一种基于深度 Q 网络(DQN)的算法来解决该优化问题。本文的主要贡献点包括以下三个方面：

a) 提出了一种基于任务优先级的“车-边-云”协同任务卸载模型，其需要通过联合优化车辆计算能力与任务卸载策略以获取系统最小延迟和能耗。

b) 该问题是一个混合整数线性规划问题，是 NP-hard 的。因此，为了降低算法复杂度，本文分两步对其进行求解。首先，通过数学推导得出了最优车辆计算能力的解析解，然后在其数值固定条件下，基于 DQN 算法获得了最佳的任务卸

收稿日期：2022-02-17；**修回日期：**2022-03-29　**基金项目：**国家自然科学基金资助项目(61802319)

作者简介：王锦(1995-)，女，山东淄博人，硕士研究生，主要研究方向为边缘计算、网络安全；张新有(1971-)，男(通信作者)，河南人三门峡人，副教授，博士，主要研究方向为网络体系结构、分布式计算与应用、网络安全(xyzhang@swjtu.edu.cn).

载策略。

c) 进行了综合全面的仿真实验，实验结果证明了算法的可行性和优越性。

1 研究现状

任务卸载技术通过将设备计算任务卸载到边缘节点或云服务器从而解决设备计算资源不足的问题。目前，已经有不少研究人员致力于研究不同场景下多用户 MEC 任务卸载问题。根据不同的优化目标，之前的研究工作可分为如下三类：

a) 针对时延的优化。文献[8]为了解决多云环境下任务卸载过程复杂、响应时间长等问题，提出了一种基于多云协作的加权自适应惯性权重粒子群优化算法。实验结果表明，与非协同任务卸载方案相比，该方案能够有效地减少任务卸载时间。文献[9]为了加快车联网环境中任务处理效率，提出了一种基于 AHP-DQN 的任务卸载算法，该算法利用 AHP 将车辆任务进行合理划分，并根据车辆实时信道增益进行卸载决策的部署。最后通过大量的仿真实验证明了该算法能够降低任务处理时延。但该方案未对车辆计算能力进行优化。文献[10]为了解决车联网环境中同时考虑 MEC 服务器及云服务器资源分配的卸载问题，提出了一种基于 DQN 的端-边-云卸载方案。并通过大量实验证明了所提方案的优越性。但该方案并未考虑到任务分类以及任务优先级问题。

b) 针对能耗的优化。文献[11]考虑在价格合理的嵌入式系统上同时提供多种自动驾驶服务，设计了一个用于实现自动驾驶机器人和车辆服务的低功耗边缘计算系统，它可以在低能耗下成功支持多种自动驾驶服务。文献[12]考虑到车辆的异质性和路边单元的影响，建立了一个考虑异构车辆和 RSU 的网络模型，并提出了一种算法以寻找最优的资源分配策略，实验表明该算法能较好地降低能耗。

c) 针对时延和能耗的优化。文献[13]考虑了一般类型的任务计算的时间分布并分析了服务器排队延迟问题，提出了一种基于整数线性规划(ILP)的算法以优化系统的时延和服务器总能耗，实验结果证明了该算法的有效性。文献[14]针对现有的计算模型难以满足新兴应用对低延迟、高复杂度和高可靠性要求的问题，提出了一种基于服务编排的云-边缘服务器协同任务卸载算法，该算法能有效降低时延和能耗。文献[15]为了最小化资源有限设备的任务完成时延和能耗问题，考虑到设备的移动性与任务依赖关系的基础上，提出了一种基于 DQN 的多目标任务卸载算法，经过多种测试场景，验证了该算法的有效性。但该方案也未考虑到任务分类以及任务优先级问题。

以上工作为解决设备或车联网背景下关于不同优化目标的任务卸载问题提供了可行的解决方案，其中针对时延和能耗的模型给本文提供了良好的启示。然而，无人驾驶汽车电池容量的有限性不可忽略，车辆计算能力需进行优化从而降低无人驾驶汽车的能耗。再者，无人车的高速移动性会导致汽车在行驶过程中超出 MEC 服务器的 V2I 通信范围。最后，无人驾驶汽车在行驶中会产生各种类型的任务，如与行驶有关的控制指令任务或与娱乐相关的任务，不同任务的紧迫性有所不同。关于设备的相关文献大多未考虑到车辆的移动性问题，关于车联网的相关文献大多未考虑到任务按照优先级分类或未对车辆的计算能力进行优化。因此，以上卸载方案不适合无人驾驶场景，目前关于无人驾驶场景下的任务卸载问题研究较少。

本文针对三层网络架构下的无人驾驶场景，考虑了无人车的计算能力优化、移动性问题以及任务按照优先级分类等影响因素，在此基础上构建了一个新的任务卸载模型，仿真结果验证了该模型的合理性和可行性。

2 系统模型

本节首先介绍了无人车场景下任务卸载的网络模型、通信模型、计算模型和优先级模型。在此基础上，定义了一个通过联合优化车辆计算能力与任务卸载策略以获取系统最小延迟和能耗的优化问题。

2.1 网络模型

本系统包括三层网络架构，由车辆层、边缘层和云层组成，如图 1 所示。车辆层包括了所有在道路上行驶的无人车。假设每一辆无人车配备了双无线电接口:802.11p 接口(支持 V2X 通信)和蜂窝网络接口(例如 LTE-A)[16]，车辆可以通过 802.11p 接口与 RSU 进行通信。边缘层由位于道路旁的边缘节点组成，其中每个边缘节点包括 RSU 以及配备在 RSU 旁的 MEC 服务器。RSU 具有一定的通信覆盖能力，可以用来收集车辆、网络状况以及任务的相关信息。云层包含云服务器，无人车通过蜂窝网络接口与基站进行通信，基站通过有线链路与云服务器进行连接，从而将无人车的任务卸载到云服务器上。

系统中存在着多辆有计算任务的无人车，MEC 服务器按照一定的时间周期为所有车辆进行卸载调度。RSU 集合定义为 K={1,2,⋯,k}，无人车集合定义为 N={1,2,⋯,n}。每一辆无人车都有计算密集型任务需要及时处理，且假设计算任务是不可分的。因此，每一个任务要么在本地执行，要么被卸载到 MEC 服务器或云服务器上去执行。当任务被卸载到 MEC 服务器时，由于车辆的移动性，在无人车驶出 RSU 的无线覆盖范围并且任务没有计算完成的情况下，其必须被卸载到云服务器上运行。为了便于表述，表 1 列出了系统模型的部分符号以及含义。

表 1 关键符号

Tab. 1 Key symbols

符号	含义
N	无人车集合
K	RSU 集合
D	任务数据大小
C	完成任务所需的 CPU 周期数
τ	最大容忍延迟
A	卸载决策
p	车辆的传输功率
B_m	MEC 服务器的上行信道总带宽
$B_{n,c}$	无人车与云服务器之间的上行信道带宽
$v_{n,m}$	车辆 n 到 MEC 服务器的传输速率
$v_{n,c}$	车辆 n 到云服务器的传输速率
g_m^2	无人车与 MEC 服务器之间的信道增益
g_c^2	无人车与云服务器之间的信道增益
N_0	噪声功率
f_n	第 n 辆无人车的计算能力
f_m	MEC 服务器的计算能力
f_c	云服务器的计算能力
P	车辆任务的优先级别

2.2 通信模型

首先引入通信模型。无人车通过无线信道将任务传输到 MEC 服务器。根据香农公式，车辆 n 到 MEC 服务器的传输速率为

$$v_{n,m}=\frac{B_m}{I}\log_2\left(1+\frac{p_n g_m^2}{\frac{B_m}{I}N_0}\right) \tag{1}$$

p_n 是指车辆 n 的传输功率。B_m 是指 MEC 服务器的上行信道总带宽。N_0 是指噪声功率。g_m^2 是指无人车与 MEC 服务

器之间的信道增益。I 是指卸载到 MEC 服务器上的任务数。

无人车通过无线信道将任务卸载到云服务器。根据香农公式，车辆 n 到云服务器的传输速率：

$$v_{n,c} = B_{n,c} \log_2\left(1 + \frac{p_n g_c^2}{B_{n,c} N_0}\right) \tag{2}$$

$B_{n,c}$ 是指无人车与云服务器之间的上行信道带宽。N_0 是指噪声功率。g_c^2 是指无人车与云服务器之间的信道增益。

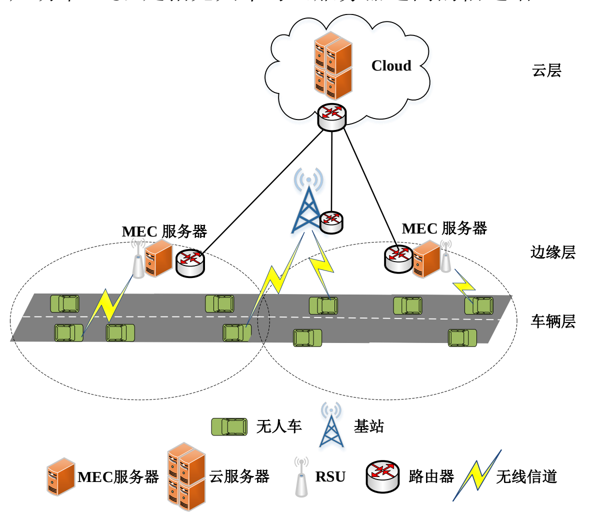


图 1 网络模型

Fig. 1 Network model

2.3 计算模型

对于模型中第 n 辆车产生的计算任务 R_n，可以用一个三元组 $R_n = \{D_n, C_n, \tau_n\}$ 进行描述。其中，D_n 表示无人车 n 的任务数据大小，C_n 表示完成任务所需的 CPU 周期数，τ_n 表示 R_n 的最大容忍延迟。设 A 为卸载决策，$A = [a_1, a_2, \ldots, a_N]$。其中 $a_n = \{a_n^l, a_n^m, a_n^c\}$，$a_n^l, a_n^m, a_n^c \in \{0,1\}$ 表示 R_n 的卸载选择，其中 a_n^l, a_n^m, a_n^c 分别表示 R_n 是否在本地、MEC 服务器或云服务器上运行，若值为 1，则代表在其上运行，反之则不是。并且任务只能在本地、MEC 服务器和云服务器中的一个执行，因此，需要满足约束 $a_n^l + a_n^m + a_n^c = 1$。

2.3.1 本地计算模型

当 $a_n^l = 1$ 时，表示任务在本地执行。用 $f_n \in [0, f_{max}]$ 表示第 n 辆无人车的计算能力，可以用 CPU 的周期频率进行描述 (CPU 周期/s)。通过应用动态电压和频率缩放技术(DVFS)[17]，无人车可以通过调整自身每个周期的 CPU 频率来平衡执行时间和能耗。为了简单起见，假设 f_n 在一个调度周期内保持不变。因此，任务在本地的执行时间为

$$T_{loc_n} = \frac{C_n}{f_n} \tag{3}$$

根据参考文献[18]，能耗与频率的平方成正比。因此，任务在本地运行所产生的能耗为

$$E_{loc_n} = kC_n (f_n)^2 \tag{4}$$

k 是功率转换系数，取决于微型集成电路架构，通常取 $k = 10^{-27}$。

2.3.2 MEC 计算模型

当 $a_n^m = 1$ 时，表示任务被卸载到 MEC 服务器上运行，其对应的传输时间为

$$T_{tra_{n,m}} = \frac{D_n}{v_{n,m}} \tag{5}$$

假设任务的计算结果非常小，因此从 MEC 服务器把计算结果返回给无人车的时延和能耗通常忽略不计[19,20]。

任务在 MEC 服务器上的执行时延为

$$T_{exe_m} = \frac{C_n}{f_m} \tag{6}$$

用 f_m 表示 MEC 服务器的计算能力。

假设 MEC 服务器是单核的，一次只能执行一个任务，所以卸载到 MEC 服务器上的任务需要排队。当一个任务到达 MEC 服务器时，它将首先被添加到等待队列中。在队列中，任务按照优先级进行排序，相同优先级的任务按照先后顺序进行排序。

本文用 T_{ts} 表示任务的数据传输开始时刻，用 T_{te} 表示任务的数据传输结束时刻。用 T_{cs} 表示任务的计算开始时刻，用 T_{ce} 表示任务的计算完成时刻。因此可以得到：

$$T_{te} = T_{ts} + T_{tra_{n,m}} \tag{7}$$

$$T_{ce} = T_{cs} + T_{exe_m} \tag{8}$$

只有前一个任务计算完成并且当前任务传输结束时，当前任务才能开始计算。公示表示如下：

$$T_{cs} = \max\{T_{pce}, T_{te}\} \tag{9}$$

其中 T_{pce} 是指前一个任务的计算完成时刻。任务在 MEC 服务器上的等待时间为计算开始时刻减去传输结束时刻，用如下公式表示：

$$T_{wait_m} = T_{cs} - T_{te} \tag{10}$$

因此，当任务被卸载到 MEC 服务器上执行时，其花费的总时间为传输时间加上执行时间再加等待时间，用如下公式表示：

$$T_{mec_n} = T_{tra_{n,m}} + T_{exe_m} + T_{wait_m} \tag{11}$$

由于车辆的移动性，完成任务所花费的总时间不能超过无人车 n 与 MEC 服务器之间的连接时间 $T_{v,m}$，公式表示如下：

$$T_{mec_n} \le T_{v,m} \tag{12}$$

本文优化的是无人车所产生的能耗，MEC 服务器所产生的能耗不予考虑。因此，不考虑任务在 MEC 服务器上执行的能耗，只考虑无人车进行任务传输时产生的能耗，因此总能耗用如下公式表示：

$$E_{mec_n} = p_n T_{tra_{n,m}} \tag{13}$$

2.3.3 云计算模型

a) 直接在云服务器执行的情况

当 $a_n^c = 1$ 时，表示在云服务器上执行此任务，其被认为拥有无限的资源。因此，忽略任务在云服务器上的排队，只考虑对应的传输时间、传输能耗和在其上的计算时间。

在云服务器上的传输时间如下：

$$T_{tra_{n,c}} = \frac{D_n}{v_{n,c}} \tag{14}$$

在云服务器上的执行时间为

$$T_{exe_c} = \frac{C_n}{f_c} \tag{15}$$

用 f_c 表示云服务器的计算能力。

所以，在云服务器上花费的总时延为

$$T_{cloud_n} = T_{tra_{n,c}} + T_{exe_c} \tag{16}$$

云服务器产生的能耗依旧不予考虑，因此在云服务器上执行产生的能耗仅为传输能耗，即：

$$E_{cloud_n} = p_n T_{tra_{n,c}} \tag{17}$$

b) MEC 服务器上执行未完成需要卸载到云服务器的情况

MEC 服务器作为计算服务器，可以通过无线 V2I 通信处理任务。由于车辆的移动性，必须在有限的 V2I 连接时间内完成任务的计算。当任务无法在车辆与 MEC 服务器的连接时间内计算完成时，将其传输到云服务器上重新执行，而且需要加上之前在 MEC 服务器上浪费的时延和能耗。

关于能耗方面，因本文只对无人车的能耗进行优化，MEC 服务器执行任务所产生的能耗不予考虑，因此在此情况浪费的能耗仅为无人车将任务传输到 MEC 服务器这个过程所产生的能耗，而能耗与时间有关，具体无人车传输任务所花费的时间存在以下几种情况。a) 未将任务传输完毕，

已跑出当前 V2I 的通信范围区域，与当前服务器的连接已断开，此时任务在 MEC 服务器上浪费的时间为连接时间。b) 将任务全部传输到 MEC 服务器时，此时无人车超出 V2I 通信范围，MEC 服务器未开始进行计算，任务在 MEC 服务器上浪费的时间为传输时间也是连接时间。c) 将任务全部传输到 MEC 服务器，MEC 服务器已经开始计算，但是计算未完成，此时无人车超出 V2I 通信范围，任务在 MEC 服务器上浪费的时间为传输时间。因此，任务在 MEC 服务器上执行浪费的时间为传输时间和连接时间的最小值：

$$T_{extra}=\min\{T_{tra_{n,m}},T_{v,m}\} \tag{18}$$

所以在 MEC 服务器上浪费的能耗用如下公式表示：

$$E_{extra}=p_nT_{extra} \tag{19}$$

在云服务器上花费的总时延为与 MEC 服务器的连接时间和重新传输给云服务器消耗的时间之和，用如下公式进行表示：

$$T'_{mec_n}=T_{v,m}+T_{tra_{n,c}}+T_{exe_c} \tag{20}$$

在云服务器上花费的总能耗为浪费的能耗与重新传输产生的能耗之和，公式表示如下：

$$E'_{mec_n}=E_{extra}+E_{cloud_n} \tag{21}$$

2.4　优先级模型

本系统在任务卸载中考虑了任务的紧迫性和完成率，以满足无人驾驶的性能需要。也就是说，它决定了如何优先处理具有较高紧迫性的任务，以及如何尽可能多地完成任务。鉴于无人驾驶汽车任务的特殊性，不同的任务具有不同的紧迫性，因此需要对它们进行分类。本文将这些任务根据其紧迫性分为三个级别，包括非常重要任务、重要任务和一般重要任务[21]，如表 2 所示。对于非常重要任务，通常其数据量较小且容忍时延较短[21]。对于重要任务，通常其数据量较大且容忍时延较长。对于一般重要任务，通常具备很高的时延容忍性。

表 2　车辆任务的优先级分类

Tab. 2　Prioritization of vehicular tasks

组别	举例	特性	优先级
非常重要任务	路径规划、防撞	数据量小、容忍时延短	1
重要任务	导航、信息服务	数据量较大、容忍时延较长	2
一般重要任务	音乐、视频、游戏	数据量大、容忍时延长	3

对于非常重要任务，需要首先为其提供服务，而对于一般重要任务，可以最后为其提供服务。定义车辆 n 的任务优先级参数为 $P_n, P_n\in\{1,2,3\}$ 。不同的车辆任务拥有不同级别的优先级。在 MEC 服务器队列中，本文采用非抢占式优先级调度算法为队列中的任务进行调度。

2.5　问题的公式化

本文将完成任务所花费的时延和能耗的加权和定义为任务的总开销。对于任务 R_n，其时延成本表述如下：

$$T_n=a_n^lT_{loc_n}+a_n^m(mT_{mec_n}+(1-m)T'_{mec_n})+a_n^cT_{cloud_n} \tag{22}$$

同样，能耗成本可以表述为

$$E_n=a_n^lE_{loc_n}+a_n^m(mE_{mec_n}+(1-m)E'_{mec_n})+a_n^cE_{cloud_n} \tag{23}$$

因此，任务的总开销为：$Z_n=\beta_n^TT_n+\beta_n^EE_n$，其中 $0\le\beta_n^T\le1, 0\le\beta_n^E\le1, \beta_n^T+\beta_n^E=1$。$\beta_n^T$ 和 β_n^E 是分配给时延和能耗的权重系数，$m=1$ 表示任务卸载到 MEC 服务器并且能够在 V2I 连接时间内顺利完成任务。

为了满足不同用户的特定需求，可以为其选择不同的时延能耗权重系数。例如：电池容量较小的无人车会选择更大的 β_n^E 来节约更多地资源；而电池容量大的无人车更倾向于选择更大的 β_n^T 来减小延迟，获得更好地体验。

该系统在一次调度周期内的优化问题可以用如下公式表示，其中 A 代表所有任务的卸载选择，F 代表所有车辆计算能力的集合。

$$\begin{aligned}&G=\min_{A,F}\sum_{n=1}^{N}Z_n\\&C1:\ a_n^l,a_n^m,a_n^c\in\{0,1\},a_n^l+a_n^m+a_n^c=1\\&C2:\ T_n\le\tau_n,\forall n\in N\\&C3:\ 0\le f_n\le f_{max},\forall n\in N\\&C4:\ m\in\{0,1\}\\&C5:\ P_n\in\{1,2,3\}\end{aligned} \tag{24}$$

其中 C1 表示任务在本地、MEC 服务器或者云服务器上执行；C2 表示完成任务 R_n 的总时延不能超过任务的最大容忍时延 τ_n；C3 表示车辆 n 的计算能力不能超过其最大计算能力；C4 表示任务是否在 MEC 服务器上顺利完成；C5 表示任务的优先级等级。

3　算法设计

问题(24)可以通过寻找最佳的卸载决策变量 A 和车辆的计算能力 F 来解决。由于卸载决策变量 A 是一个离散变量集合，而车辆的计算能力 F 是连续变量集合，因此该问题为混合整数非线性规划问题，是 NP-hard 的。针对此问题，本文采用分步求解来解决。

3.1　车辆最佳计算能力的求解

在卸载决策即 A 固定的情况下，优化目标 G 中的变量只剩下车辆的计算能力 F，而 F 只会影响选择在本地运行的任务，在 MEC 服务器和云服务器中运行的任务所产生的时延和能耗可看做常数 C。因此，问题(24)中的优化模型可以变为

$$\begin{aligned}&G=\min_{F}\sum_{n=1}^{N}(\beta_n^TT_{local_n}+\beta_n^EE_{local_n})+C=\\&\min_{F}\sum_{n=1}^{N}\left(\beta_n^T\frac{C_n}{f_n}+\beta_n^EkC_n(f_n)^2\right)+C\\&T_{local_n}\le\tau_n\\&0\le f_n\le f_{max},\forall n\in N\end{aligned} \tag{25}$$

由这两个约束条件又可推出：

$$\frac{C_n}{\tau_n}\le f_n\le f_{max} \tag{26}$$

对 G 关于 f_n 进行求导并令 $G'=0$，可以求出对应的 f_n，如下所示。

$$f_n^*=\sqrt[3]{\frac{\beta_n^T}{2\beta_n^Ek}} \tag{27}$$

显然，G 在 $[0,f_n^*]$ 内单调递减，在 $[f_n^*,+\infty]$ 内单调递增，所以在不考虑约束条件即式(26)时，f_n^* 为车辆的最优计算能力。可以观察到，车辆的最优计算能力随着延迟权重的增加而变大。

但因为存在约束，且车辆的计算能力不一定能满足任务的实时性要求，因此需要分情况讨论：

a) 当 $\frac{C_n}{\tau_n}>f_{max}$ 时

(a) 当 $f_n^*>f_{max}$ 时，$f_n=f_{max}$ 时，G 最小。

(b) 当 $f_n^*\le f_{max}$ 时，$f_n=f_n^*$ 时，G 最小。

b) 当 $\frac{C_n}{\tau_n}\le f_{max}$ 时

(a) 当 $f_n^*\le\frac{C_n}{\tau_n}$ 时，此时 G 在 $\left[\frac{C_n}{\tau_n},f_{max}\right]$ 单调递增，所以 $f_n=\frac{C_n}{\tau_n}$ 时，G 最小。

(b) 当 $\frac{C_n}{\tau_n}\le f_n^*\le f_{max}$ 时，此时 $f_n=f_n^*$ 时，G 最小。

(c) 当 $f_n^*\ge f_{max}$ 时，此时 G 在 $\left[\frac{C_n}{\tau_n},f_{max}\right]$ 单调递减，所以 $f_n=f_{max}$ 时，G 最小。

3.2　基于 DQN 的任务卸载算法

本小节设计了一种基于 DQN 的算法来解决无人驾驶汽车的任务卸载决策问题。在本场景中，DQN 的三要素包括状态、动作和奖励函数。

a) 状态空间：状态是用来反映环境的，它由车辆、任务和 MEC 服务器的相关信息组成。用 $S=\{s_1,s_2,s_3,\ldots,s_n,\ldots\}$ 表示整个系统的状态空间，s_n 为第 n 辆车的系统状态，$s_n=\{f_{max},p_n,D_n,C_n,\tau_n,B_m,f_m,M_{load_n}\}$，其中 f_{max} 和 p_n 分别表示车辆的最大计算能力和传输速率，D_n,C_n,τ_n 分别表示任务的数据量、所需计算资源和最大容忍延迟，B_m,f_m,M_{load_n} 分别表示MEC服务器的上行信道总带宽、MEC 服务器的计算能力和 MEC 服务器当前负载情况。因为 MEC 服务器依次为所有任务制定卸载决策，并且之前任务的卸载选择会影响之后任务的卸载选择，所以本节将 MEC 服务器当前负载情况也作为状态的一部分。

b) 动作空间：动作表示任务的卸载决策，即任务在本地、MEC 服务器或者云服务器上执行的选择。定义系统的动作空间为 $A=\{a_1,a_2,a_3,\ldots,a_n,\ldots\}$ 。$a_n=\{a_n^l,a_n^m,a_n^c\}$，其中 a_n^l,a_n^m,a_n^c 分别表示任务在本地、MEC 服务器和云服务器的卸载决策。

c) 奖励函数：agent 执行一个动作 a_n 之后，会在该动作对应的状态 s_n 下获得奖励 r_n 。一般来说，奖励函数和优化目标有关，且本文的优化目标是最小化系统的总成本，因此考虑奖励函数 r_n 与 $-Z_n$ 相关。此外，任务的完成率也是一个很重要的性能指标，因此，本文将其也作为奖励函数的一部分。当任务未能在最大容忍延迟内完成时，应该对它进行惩罚，因此最终奖励函数定义为

$$r_n=\begin{cases}-(\beta_n^T T_n+\beta_n^E E_n) & T_n\le\tau_n\\ -(\beta_n^T T_n+\beta_n^E E_n)-(T_n-\tau_n) & T_n>\tau_n\end{cases} \tag{28}$$

在定义了以上三个关键要素之后，基于 DQN 的无人驾驶场景任务卸载过程如下：MEC服务器为DQN算法的agent，其通过状态、动作和奖励与环境进行交互。当 agent 收到车辆的卸载请求时，将根据当前环境状态为该车辆寻找最佳卸载决策，即动作，之后将动作的取值返回给车辆。车辆执行动作后，agent 会收到执行该动作得到的奖励。

传统的 Q-learning 算法使用 Q-table 来存储每个时刻的状态、动作下所获得的 Q 值 $Q(s,a)$，agent 通过遍历 Q-table，寻找在当前状态下可以获得最大奖励的动作。然而，随着状态空间和动作空间的增加，存储 Q-table 所需的内存空间迅速增加，因此会导致更新和搜索 Q-table 时花费很大的时间开销。

本文使用 DQN 算法来解决上述问题。DQN 采用神经网络来估计不同状态-动作组合下的 Q 值。为了打破数据的关联性以及保持训练的稳定性，DQN 算法加入了经验回放技术以及双网络机制。经验回放是指将 agent 与环境互动过程中产生的一条条数据存入到经验池D中，并在后面的训练中从经验池D中随机采样小批量数据进行网络的更新。双网络是指通过引入评价网络 Q^{eva} 以及目标网络 Q^{tar}，评价网络和目标网络的参数分别为 θ^{eva} 和 θ^{tar} 。其中评价网络用来评估当前状态下每一个动作的Q值，目标网络用来计算下一个状态对应动作的 Q 值。首先，随机初始化 θ^{eva} 和 θ^{tar}，并创建一个容量大小确定的经验池 D。对于 episode 内的当前 step n，算法将状态 s_n 输入到评价网络中以获得每个动作对应的概率，并根据动作概率使用 Categorical 函数选择动作 a_n，得到对应奖励 r_n 。之后环境进入下一个状态 s_{n+1}，并将经验轨迹 (s_n,a_n,r_n,s_{n+1}) 作为一条数据存入经验池 D 中。最后，从经验池D中随机选择小批量的数据进行评价网络的更新，并在C步迭代后，将评价网络的参数同步给目标网络。其中，目标 Q 网络的值的计算公式如下所示。

$$Q^t=r_n+\gamma\max_{a'}Q^{tar}(s_{n+1},a';\theta^{tar}) \tag{29}$$

使用均方误差[22]作为算法损失函数，并通过梯度下降算法[23]来最小化该损失，损失函数的定义如下所示。

$$Loss=\frac{1}{2}(Q^t-Q^{eva}(s,a;\theta^{eva}))^2 \tag{30}$$

以下为 DQN 算法的详细步骤：

算法 1　DQN 算法

输入：评价网络 Q^{eva}，参数 θ^{eva}，目标网络 Q^{tar}，参数 θ^{tar}，经验池 D，当前状态 s_n

输出：动作 a_n

随机初始化评价网络和目标网络的参数 θ^{eva} 和 θ^{tar}，$\theta^{eva}=\theta^{tar}$
初始化大小为 N 的经验池 D
　for each episode k=1 to K do
　　for each step n do
　　　获取当前的状态 s_n
　　　通过 Categorical 函数获取当前的动作
　　　计算奖励 r_n 并观察下一个状态 s_{n+1}
　　　把 (s_n,a_n,r_n,s_{n+1}) 作为一条数据存入经验池 D 中
　　　从经验池 D 中随机选取小批量的数据
　　　if　s_{n+1} 是最终状态 then
　　　　$Q^t=r_n$
　　　else
　　　　$Q^t=r_n+\gamma\max_{a'}Q^{tar}(s_{n+1},a';\theta^{tar})$
　　　end if
　　　根据式(30)计算损失函数 Loss，并使用梯度下降法更新评价网络的参数 θ^{eva}
　　　C 轮迭代之后复制参数 θ^{eva} 来更新目标网络参数 θ^{tar}
　　end for
　end for

4　性能评估

在本节中，首先对实验平台以及数据集进行简要说明。本实验使用 Intel(R) Core(TM) i5-10210U CPU，16GB RAM，1.6GHz 的个人计算机开发，所有的实验环境均使用 python3.7 和 pytorch1.5.0 实现。地图取自西安市区 5km*5km 大小的区域，车辆轨迹由开源道路交通仿真软件 SUMO[24] 生成。

4.1　参数设置

实验中，设置了 1 个 MEC 服务器，其 V2I 的通信半径为 500 米。无人车的数量随着时间的推移逐渐增多，每一辆无人车在一个调度周期内至多产生一个任务，任务产生概率为 0.6。任务有三种不同的优先级，参照文献[25-27]对三种任务进行参数设置。其中，第一优先级任务到第三优先级任务的数据量大小分别在[200,300]KB、[400,500]KB 和[700,800]KB 内随机选取，所需要的计算资源分别在[50,80]M CPU cycles、[150,180]M CPU cycles 和[250,280]M CPU cycles 内随机选取，最大容忍时延分别为 150ms、200ms 和 350ms。无人车的最大计算能力为 1.5G CPU cycle/s，传输功率为 250mW。信道增益在[1,2]范围内随机选取。MEC 服务器的带宽为 50MHz，计算能力为 4.5GHz。云服务器的带宽为 10MHz，计算能力为 1.5GHz。除此之外，MEC 服务器每隔 2s 为车辆进行一次任务卸载调度。

在算法实现方面，参考文献[28]，DQN 算法的网络结构和超参数设置为：Q 网络的层数为 5 层，隐藏层神经元数量为 128。学习率和折扣系数分别为 0.01 和 0.95。经验池的大小设置为 100，batch 大小为 20。

4.2　对比实验

本文选择全本地、全 MEC 服务器和随机三种常用的任务卸载算法与 DQN 算法进行对比，并将目标值(时延和能耗的总成本)、时延、能耗以及完成率作为评价指标。三种对比算法的介绍如下：

a) 全本地执行：任务全部在无人车本地执行。

b) 全 MEC 服务器执行：任务全部被卸载到 MEC 服务器上执行。

c) 随机卸载：任务随机选择在无人车本地、MEC 服务器或云服务器上执行。

4.3　仿真结果

图 2 显示了调度次数为 18500 次时累计平均奖励的收敛曲线图，可以看出，在一定量的训练步数之后，基于 DQN 的任务卸载算法能够实现稳定的收敛。原因之一是 DQN 算法在与环境的多次交互中，充分对环境进行了探索，并在此基础上学习到了适合的任务卸载策略。另外一个原因则是经验池的使用能够有效地降低样本数据之间的关联性，提高算法的稳定性，使算法更容易收敛。

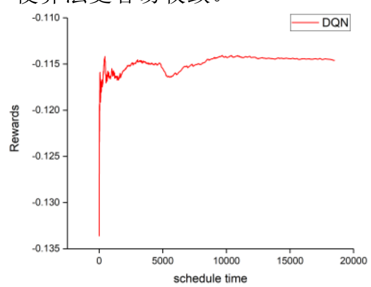
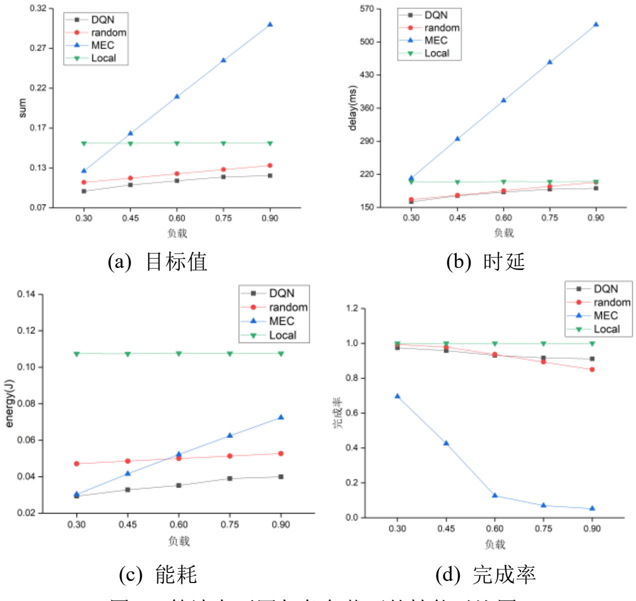


图 2　累计平均奖励收敛曲线图

Fig. 2　Convergence Curve of Cumulative Average Reward

图 3 通过改变每一辆无人车任务产生的概率，比较了 DQN 算法和其他三种对比算法在不同任务负载下的性能，其中，无人车请求任务卸载的概率在 0.3 到 0.9 之间变化。从图中可以看出，除全本地执行外，随着任务负载的增加，其他三种算法的目标值、时延和能耗都会变大并且完成率降低。然而，由于全本地执行与任务卸载概率的关联性不大，所以评价指标接近一条水平线。此外，在所有的算法中，DQN 算法的目标值、延迟以及能耗都是最小的，并且其完成率也较高，与全本地执行相差不大。

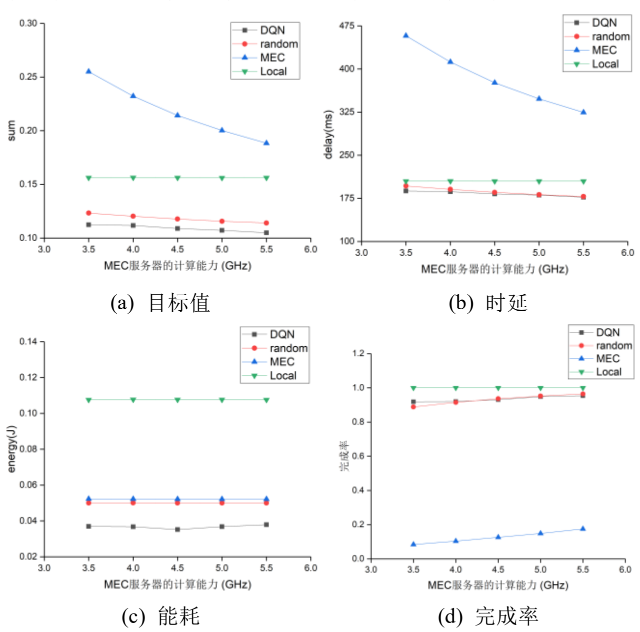


(a) 目标值　(b) 时延　(c) 能耗　(d) 完成率

图 3　算法在不同任务负载下的性能对比图

Fig. 3　Performance comparison of algorithms under different task loads

图 4 比较了四种算法在不同 MEC 服务器计算能力下的性能。通过观察，全本地执行的四种评价指标持一条水平线，这是因为 MEC 服务器计算能力的变化不会对无人车的计算能力产生影响。随着 MEC 服务器计算能力的增大，其他三种算法的目标值和延迟随之减小并且完成率增加，而能耗与 MEC 服务器的计算能力无关，所以基本呈一条水平线。此外，在所有的策略中，DQN 算法的目标值、延迟以及能耗都是最小的，其完成率表现也较好，与全本地执行相差不大。

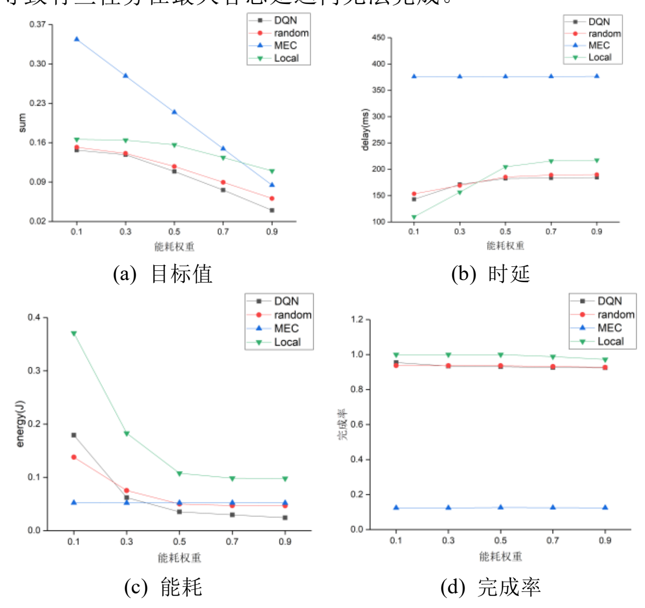


(a) 目标值　(b) 时延　(c) 能耗　(d) 完成率

图 4　算法在不同 MEC 服务器计算能力下的性能对比图

Fig. 4　Performance comparison of algorithms under different computation capabilities of MEC server

图 5 比较了能耗权重系数对评价指标的影响。从图中可以看出，无论在哪种权重情况下，DQN 算法的目标值都是最小的。随着能耗权重系数的增大，DQN 算法、随机算法与全本地算法的时延逐渐增大，能耗逐渐降低，说明 DVFS 技术倾向于减弱车辆的计算能力，从而降低车辆的能耗。然而，由于全 MEC 服务器执行不受车辆计算能力的影响，所以其时延、能耗和完成率接近一条水平线。此外，随着能耗权重系数的增大，全本地执行的完成率较其他三种策略降低明显，这是因为车辆计算能力的取值与延迟能耗的权重系数有关。当能耗权重系数增大时，车辆计算能力会随之减弱，导致有些任务在最大容忍延迟内无法完成。



(a) 目标值　(b) 时延　(c) 能耗　(d) 完成率

图 5　算法在不同能耗权重系数下的性能对比图

Fig. 5　Performance comparison of algorithms under different weight coefficients of energy consumption

5 结束语

本文考虑到车辆的计算能力优化、移动性问题以及任务的优先级分类等因素，提出了一个无人驾驶场景下的基于车-边-云三层网络架构的任务卸载模型，其需要通过联合优化车辆计算能力与任务卸载策略以获取系统最小延迟和能耗。由于该问题是一个混合整数非线性规划问题，为了降低算法的时间复杂度，本文采用分步求解，首先推导出了车辆计算能力的最优值，然后使用 DQN 算法来获得任务的最佳卸载决策。最后，通过大量的仿真实验验证了所提策略的可行性和优越性。在未来的工作中，将考虑相邻 RSU 之间的切换问题，另外将缓存技术加入到任务卸载架构中，以进一步降低任务的处理时延和计算能耗。

参考文献：

[1] Hee L G, Faundorfer F, Pollefeys M. Motion estimation for self-driving cars with a generalized camera [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013: 2746-2753.

[2] Ackerman E. Lidar that will make self-driving cars affordable [News] [J]. IEEE Spectrum, 2016, 53 (10): 14-14.

[3] Ning Z, Xia F, Ullah N, *et al.* Vehicular social networks: enabling smart mobility [J]. IEEE Communications Magazine, 2017, 55 (5): 16-55.

[4] Chen L, Englund C. Cooperative intersection management: A survey [J]. IEEE Trans on Intelligent Transportation Systems, 2015, 17 (2): 570-586.

[5] Zhang W, Zhang Z, Chao H C. Cooperative fog computing for dealing with big data in the internet of vehicles: architecture and hierarchical resource management [J]. IEEE Communications Magazine, 2017, 55 (12): 60-67.

[6] Weisong S, Hui S, Jie C, *et al.* Edge computing—An emerging computing model for the Internet of everything era [J]. Journal of Computer Research and Development, 2017, 54 (5): 907.

[7] 吕品，许嘉，李陶深，等. 面向自动驾驶的边缘计算技术研究综述 [J]. 通信学报, 2021, 42 (3): 190-208. (Lyu Pin, Xu Jia, Li Taoshen, *et al.* A review of edge computing technology research for autonomous driving [J]. Journal on Communications, 2021, 42 (3): 190-208.)

[8] Wang Q, Mao Y, Wang Y, *et al.* Computation tasks offloading scheme based on multi-cloudlet collaboration for edge computing [C]// the Seventh International Conference on Advanced Cloud and Big Data (CBD) . IEEE, 2019: 339-344.

[9] 赵海涛，张唐伟，陈跃，等. 基于 DQN 的车载边缘网络任务分发卸载算法 [J]. 通信学报, 2020, 41 (10): 172-178. (Zhao Haitao, Zhang Tangwei, Chen Yue, *et al.* DQN-based task distribution and offloading algorithm for in-vehicle edge network [J]. Journal on Communications, 2020, 41 (10): 172-178.)

[10] 刘国志，代飞，莫启，等. 车辆边缘计算环境下基于深度强化学习的服务卸载方法 [J/OL]. 计算机集成制造系统: 1-16. (2021-10-29) [2022-03-18]. http://kns. cnki. net/kcms/detail/11. 5946. TP. 20211029. 1534. 004. html. (Liu Guozhi, Dai Fei, Mo Qi, *et al.* Service offloading method based on deep reinforcement learning in vehicle edge computing environment [J/OL]. Computer Integrated Manufacturing Systems: 1-16. (2021-10-29) [2022-03-18]. http://kns. cnki. net/kcms/detail/11. 5946. TP. 20211029. 1534. 004. html.)

[11] Tang J, Liu S, Liu L, *et al.* Lopecs: A low-power edge computing system for real-time autonomous driving services [J]. IEEE Access, 2020, 8: 30467-30479.

[12] Lin C C, Deng D J, Yao C C. Resource allocation in vehicular cloud computing systems with heterogeneous vehicles and roadside units [J]. IEEE Internet of Things Journal, 2017, 5 (5): 3692-3700.

[13] Belogaev A, Elokhin A, Krasilov A, *et al.* Cost-effective V2X task offloading in MEC-assisted intelligent transportation systems [J]. IEEE Access, 2020, 8: 169010-169023.

[14] Huang M, Liu W, Wang T, *et al.* A cloud–MEC collaborative task offloading scheme with service orchestration [J]. IEEE Internet of Things Journal, 2019, 7 (7): 5792-5805.

[15] 邓世权，叶绪国. 基于深度 Q 网络的多目标任务卸载算法 [J/OL]. 计算机应用: 1-9. (2022-02-25) [2022-03-18]. http://kns. cnki. net/kcms/detail/51. 1307. TP. 20220223. 1728. 004. html. (Deng Shiquan, Ye Xuguo. Multi-objective task offloading algorithm based on deep Q network [J/OL]. Journal of Computer Applications: 1-9. (2022-02-25) [2022-03-18]. http://kns. cnki. net/kcms/detail/51. 1307. TP. 20220223. 1728. 004. html.)

[16] Sun Y, Xu L, Tang Y, *et al.* Traffic offloading for online video service in vehicular networks: A cooperative approach [J]. IEEE Trans on Vehicular Technology, 2018, 67 (8): 7630-7642.

[17] Qu G. What is the limit of energy saving by dynamic voltage scaling? [C]// IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers (Cat. No. 01CH37281) . IEEE, 2001: 560-563.

[18] Miettinen A P, Nurminen J K. Energy efficiency of mobile clients in cloud computing [C]// 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10) . 2010.

[19] Chen X, Jiao L, Li W, *et al.* Efficient multi-user computation offloading for mobile-edge cloud computing [J]. IEEE/ACM Trans on Networking, 2015, 24 (5): 2795-2808.

[20] Elgendy I A, Zhang W, Tian Y C, *et al.* Resource allocation and computation offloading with data security for mobile edge computing [J]. Future Generation Computer Systems, 2019, 100: 531-541.

[21] Dai H, Zeng X, Yu Z, *et al.* A scheduling algorithm for autonomous driving tasks on mobile edge computing servers [J]. Journal of Systems Architecture, 2019, 94: 14-23.

[22] Chen L, Qu H, Zhao J, *et al.* Efficient and robust deep learning with correntropy-induced loss function [J]. Neural Computing and Applications, 2016, 27 (4): 1019-1031.

[23] Ruder S. An overview of gradient descent optimization algorithms [J]. arXiv preprint arXiv: 1609. 04747, 2016.

[24] Lopez P A, Behrisch M, Bieker-Walz L, *et al.* Microscopic traffic simulation using sumo [C]// 21st International Conference on Intelligent Transportation Systems (ITSC) . IEEE, 2018: 2575-2582.

[25] Hao Z, Sun Y, Li Q, *et al.* Delay-energy efficient computation offloading and resources allocation in heterogeneous network [C]// IEEE Global Communications Conference (GLOBECOM) . IEEE, 2019: 1-6.

[26] Tran T X, Pompili D. Joint task offloading and resource allocation for multi-server mobile-edge computing networks [J]. IEEE Trans on Vehicular Technology, 2018, 68 (1): 856-868.

[27] Zhan W, Duan H, Zhu Q. Multi-user offloading and resource allocation for vehicular multi-access edge computing [C]// IEEE International Conferences on Ubiquitous Computing & Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS) . IEEE, 2019: 50-57.

[28] Elgendy I A, Zhang W Z, He H, *et al.* Joint computation offloading and task caching for multi-user and multi-task MEC systems: reinforcement learning-based algorithms [J]. Wireless Networks, 2021, 27 (3): 2023-2038.